
Making Decisions

Introduction

▷ Introduction

Utility

Decision Networks

Markov Decision
Processes

- Planning under uncertainty should address:
 - The world is nondeterministic.
 - Actions are not certain to succeed.
 - Many events are outside of the agent's control.
 - An agent doesn't know the complete state of the world.
 - An agent has multiple goals. Some could be partially satisfied.
- Use *probability* to represent ignorance.
- Use *utility* to represent preference.
- A *rational agent maximizes expected utility*.

Definitions

Introduction

Utility

▷ Definitions

Utility

Utility of Money

Money Preferences

Prospect Theory

Utility and Time

Rewards

Discount Properties

Decision Networks

Markov Decision

Processes

- An agent specifies *preferences* on *outcomes* (notation o_1, o_2, \dots).
- $o_1 \succ o_2$ means outcome o_1 is *preferred* over o_2 .
 $o_1 \sim o_2$ means *indifference* between o_1 and o_2 .
- A *lottery* is a probability distribution over outcomes, written as

$$[p_1 : o_1, p_2 : o_2, \dots, p_k : o_k]$$

where the p_i s are probabilities that sum to 1.

- The lottery specifies that outcome o_i occurs with probability p_i .
- The outcomes in a lottery can be other lotteries.

Utility

Introduction

Utility

Definitions

▷ Utility

Utility of Money

Money Preferences

Prospect Theory

Utility and Time

Rewards

Discount Properties

Decision Networks

Markov Decision
Processes

If an agent is *rational*, then there is a utility function u such that

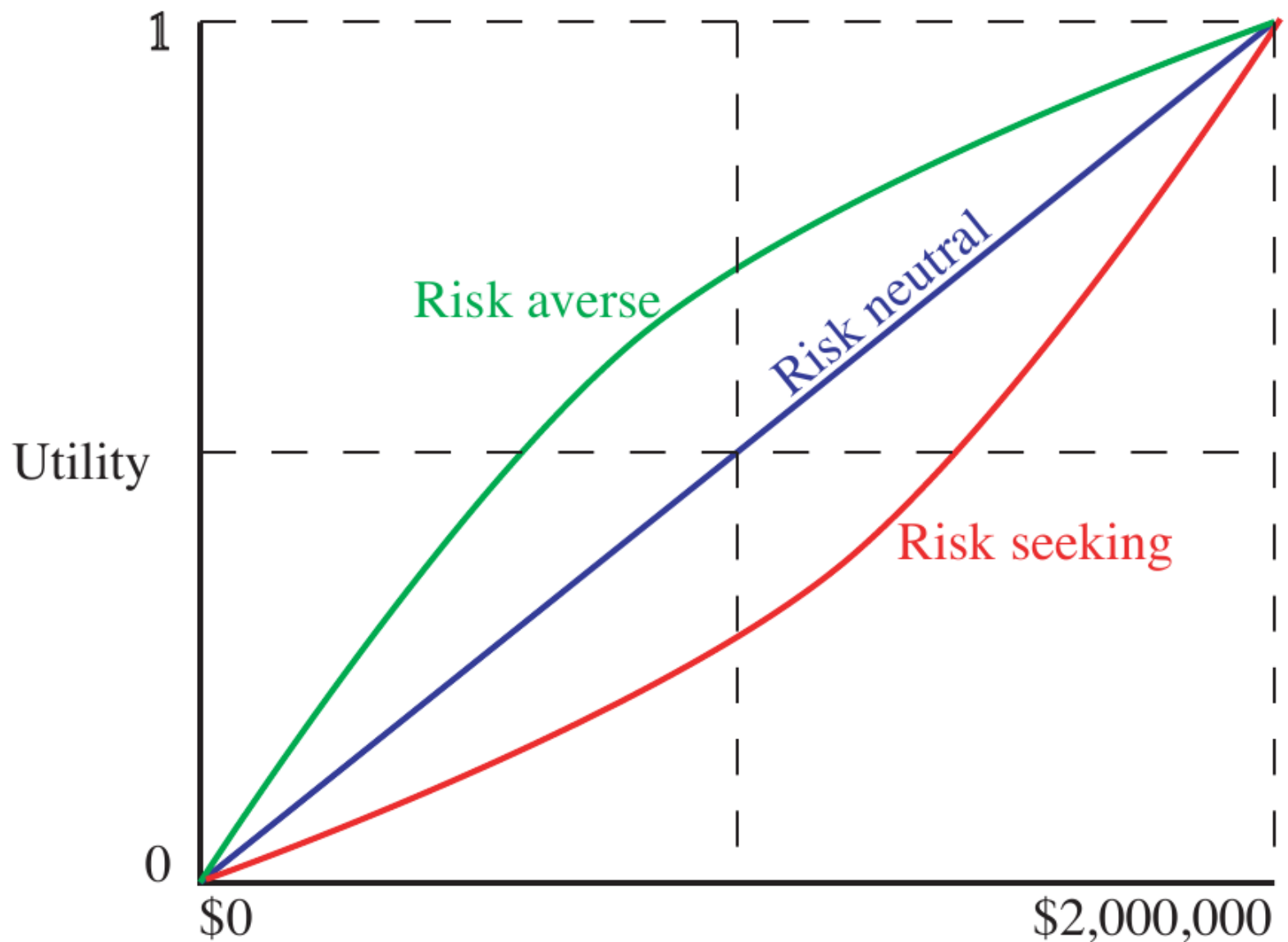
- $o_i \succ o_j$ if and only if $u(o_i) > u(o_j)$ and
- u is linear with probabilities:

$$u([p_1 : o_1, p_2 : o_2, \dots, p_k : o_k]) = p_1 u(o_1) + p_2 u(o_2) + \dots + p_k u(o_k)$$

Rational is defined as satisfying the following properties of preferences (see book):
Completeness, Transitivity, Monotonicity,
Decomposability, Continuity, Substitutability

Utility of Money

- Introduction
- Utility
- Definitions
- Utility
 - ▷ Utility of Money
- Money Preferences
- Prospect Theory
- Utility and Time
- Rewards
- Discount Properties
- Decision Networks
- Markov Decision Processes



Money Preferences

Introduction

Utility

Definitions

Utility

Utility of Money

▷ Money
Preferences

Prospect Theory

Utility and Time

Rewards

Discount Properties

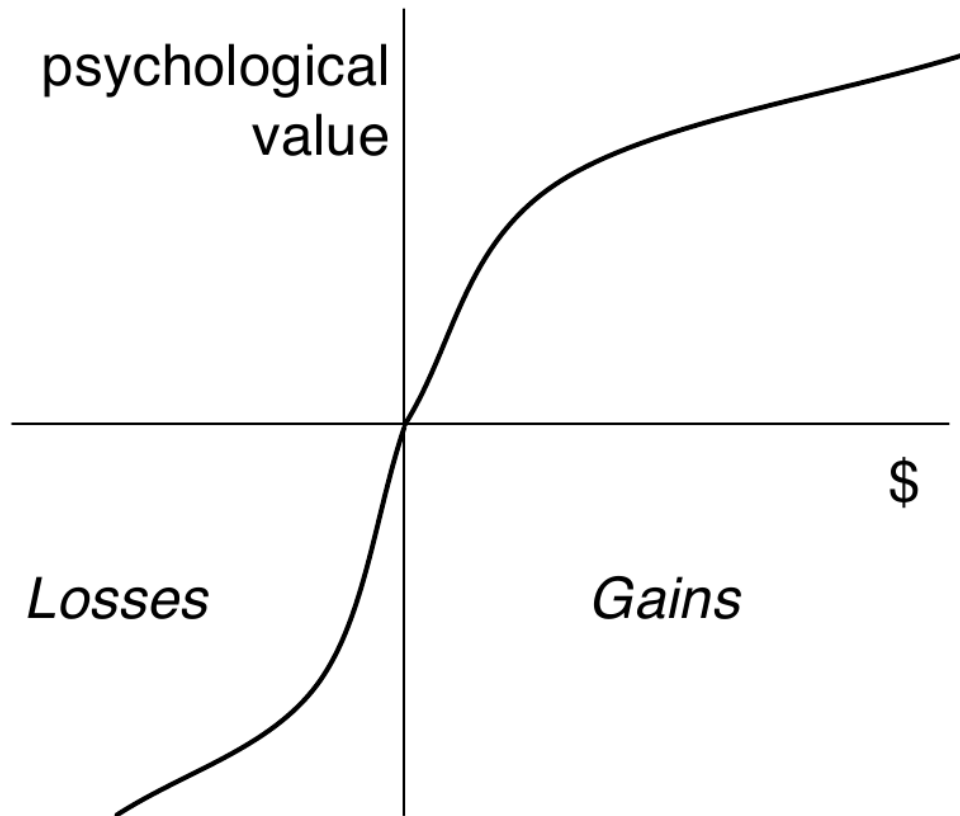
Decision Networks

Markov Decision
Processes

- What would you prefer?
 - A. \$1,000,000
 - B. lottery [0.5:\$0, 0.5:\$2,000,000]
- What would you prefer?
 - C. \$1m – one million dollars
 - D. lottery [0.10:\$2.5m, 0.89:\$1m, 0.01:\$0]
- What would you prefer?
 - E. lottery [0.11:\$1m, 0.89:\$0]
 - F. lottery [0.10:\$2.5m, 0.9:\$0]

People's Typical Value of Money

- Introduction
- Utility
- Definitions
- Utility
- Utility of Money
- Money Preferences
- ▷ Prospect Theory
- Utility and Time
- Rewards
- Discount Properties
- Decision Networks
- Markov Decision Processes



- Loss aversion is greater than gain utility.
- If all choices bad, behavior is risk seeking.

Utility and Time

Introduction

Utility

Definitions

Utility

Utility of Money

Money Preferences

Prospect Theory

▷ Utility and Time

Rewards

Discount Properties

Decision Networks

Markov Decision

Processes

- Would you prefer \$1000 today or \$1000 next year?
- How would you compare the following sequences of rewards (per day, per week, per year)?
 - A. \$1000000, \$0, \$0, \$0, \$0, \$0, ...
 - B. \$1000, \$1000, \$1000, \$1000, \$1000, ...
 - C. \$1000, \$0, \$0, \$0, \$0, ...
 - D. \$1, \$1, \$1, \$1, \$1, ...
 - E. \$1, \$2, \$3, \$4, \$5, ...

Rewards

Introduction

Utility

Definitions

Utility

Utility of Money

Money Preferences

Prospect Theory

Utility and Time

▷ Rewards

Discount Properties

Decision Networks

Markov Decision
Processes

Suppose the agent receives a sequence of rewards $r_1, r_2, r_3, r_4, \dots$ in time. What utility should be assigned?

- total reward $V = \sum_{i=1}^n r_i$
- average reward $V = \sum_{i=1}^n r_i / n$

- discounted reward

$$V = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \dots = \sum_{i=1}^n \gamma^{i-1} r_i$$

where $0 < \gamma < 1$ is the *discount factor*.

- Why discount? A reward now is worth more than the same reward later.

Properties of Discounts

Introduction

Utility

Definitions

Utility

Utility of Money

Money Preferences

Prospect Theory

Utility and Time

Rewards

▷ Discount
Properties

Decision Networks

Markov Decision
Processes

- With an infinite horizon, the total reward can be infinite: \$1, \$1, \$1, ...
and average reward can be infinite: \$1, \$2, \$3, ...
It is hard to compare one infinite with another.
- Discounted reward is always finite.
- Note that $1 + \gamma + \gamma^2 + \gamma^3 + \dots = 1/(1 - \gamma)$
- If r_{min} and r_{max} are the minimum and maximum rewards, discounted reward is between $r_{min}/(1 - \gamma)$ and $r_{max}/(1 - \gamma)$

Assumptions

Introduction

Utility

Decision Networks

▷ Assumption

Single-Stage

Example

Expected Utility

Sum Out

Sum Out Again

Sequential

Variable Types

Example

Start Sum Out

Finish Sum Out

Start Sum Out

Finish Sum Out

Finish Sum Out

Markov Decision

Processes

Decision theory for intelligent agents assumes:

- Agents know what actions they can carry out.
- The effects of actions are described as probabilities over outcomes.
- An agent's preferences are expressed by utilities of outcomes.

Decision theory specifies how to trade off the desirability and probabilities of different outcomes from different actions.

Single-State Decision Networks

Introduction

Utility

Decision Networks

Assumption

▷ Single-Stage

Example

Expected Utility

Sum Out

Sum Out Again

Sequential

Variable Types

Example

Start Sum Out

Finish Sum Out

Start Sum Out

Finish Sum Out

Finish Sum Out

Markov Decision
Processes

Single-stage decision networks extend belief networks. There are three types of variables:

- *Random variables* are belief network variables. Value depends on probability table.
- *Decision variables* whose values are actions. The agent chooses a value for each decision variable. Drawn as rectangles.
- *Utility node*. Its parents are the variables on which the utility depends. Drawn as a diamond.

Solve by calculating the *expected utility* of each decision assignment.

Example

Introduction

Utility

Decision Networks

Assumption

Single-Stage

▷ Example

Expected Utility

Sum Out

Sum Out Again

Sequential

Variable Types

Example

Start Sum Out

Finish Sum Out

Start Sum Out

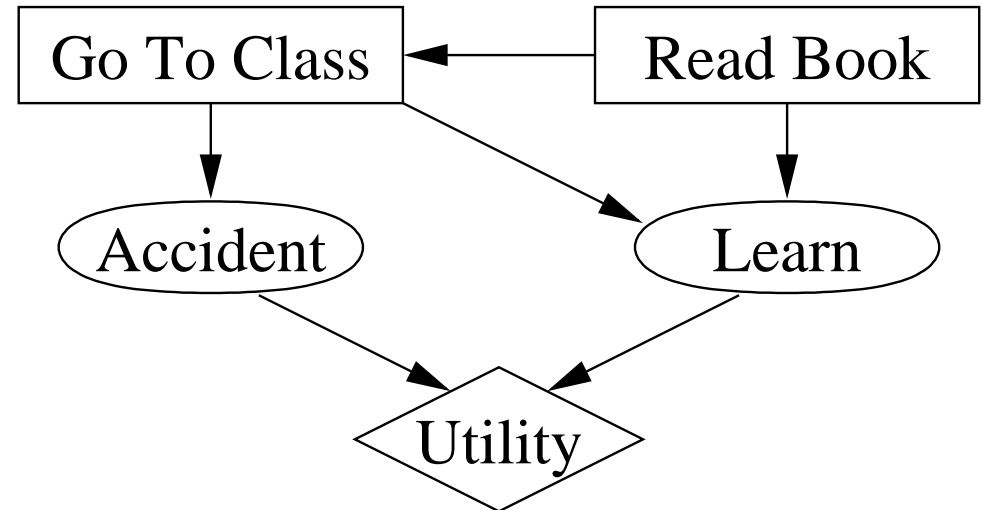
Finish Sum Out

Finish Sum Out

Markov Decision

Processes

G	$A=T$	$A=F$
T	0.001	0.999
F	0.0001	0.9999



G	R	$L=T$	$L=F$
T	T	0.9	0.1
T	F	0.5	0.5
F	T	0.5	0.5
F	F	0.1	0.9

A	L	U
T	T	-99
T	F	-100
F	T	1
F	F	0

Expected Utility

Introduction

Utility

Decision Networks

Assumption

Single-Stage

Example

▷ Expected Utility

Sum Out

Sum Out Again

Sequential

Variable Types

Example

Start Sum Out

Finish Sum Out

Start Sum Out

Finish Sum Out

Finish Sum Out

Markov Decision Processes

- A possible world ω assigns a value to each random/decision variable.
- Let the decision(s) be δ .
- Expected utility of $\delta = \sum_{\omega} P(\omega | \delta) * U(\omega)$
- Expected utility of $G = T, R = F$ is 0.4.

G	R	A	L	P	u
T	F	T	T	0.0005	-99
T	F	T	F	0.0005	-100
T	F	F	T	0.4995	1
T	F	F	F	0.4995	0

- Variable elimination can be used for expected utility. Eliminate vars below decision vars, then select best decisions.

Sum Out One Random Variable

- Introduction
- Utility

- Decision Networks
- Assumption
- Single-Stage
- Example
- Expected Utility
- ▷ Sum Out
- Sum Out Again
- Sequential
- Variable Types
- Example
- Start Sum Out
- Finish Sum Out
- Start Sum Out
- Finish Sum Out
- Finish Sum Out
- Markov Decision Processes

<i>G</i>	<i>A</i>	<i>P</i>
<i>T</i>	<i>T</i>	0.001
<i>T</i>	<i>F</i>	0.999
<i>F</i>	<i>T</i>	0.0001
<i>F</i>	<i>F</i>	0.9999

 $*$

<i>A</i>	<i>L</i>	<i>U</i>
<i>T</i>	<i>T</i>	-99
<i>T</i>	<i>F</i>	-100
<i>F</i>	<i>T</i>	1
<i>F</i>	<i>F</i>	0

Multiplying and summing out *A* yields:

<i>G</i>	<i>L</i>	<i>ExpectedUtility</i>
<i>T</i>	<i>T</i>	$0.001 * (-99) + 0.999 * (1) = 0.9$
<i>T</i>	<i>F</i>	$0.001 * (-100) + 0.999 * (0) = -0.1$
<i>F</i>	<i>T</i>	$0.0001 * (-99) + 0.9999 * (1) = 0.99$
<i>F</i>	<i>F</i>	$0.0001 * (-100) + 0.9999 * (0) = -0.01$

Sum Out The Other Random Variable

- Introduction
- Utility

- Decision Networks
- Assumption
- Single-Stage
- Example
- Expected Utility
- Sum Out
- ▷ Sum Out Again
- Sequential
- Variable Types
- Example
- Start Sum Out
- Finish Sum Out
- Start Sum Out
- Finish Sum Out
- Finish Sum Out

- Markov Decision Processes

<i>G</i>	<i>R</i>	<i>L</i>	<i>P</i>
<i>T</i>	<i>T</i>	<i>T</i>	0.9
<i>T</i>	<i>T</i>	<i>F</i>	0.1
<i>T</i>	<i>F</i>	<i>T</i>	0.5
<i>T</i>	<i>F</i>	<i>F</i>	0.5
<i>F</i>	<i>T</i>	<i>T</i>	0.5
<i>F</i>	<i>T</i>	<i>F</i>	0.5
<i>F</i>	<i>F</i>	<i>T</i>	0.1
<i>F</i>	<i>F</i>	<i>F</i>	0.9

*

<i>G</i>	<i>L</i>	<i>EU</i>
<i>T</i>	<i>T</i>	0.9
<i>T</i>	<i>F</i>	-0.1
<i>F</i>	<i>T</i>	0.99
<i>F</i>	<i>F</i>	-0.01

Multiply and sum out *L*:

<i>G</i>	<i>R</i>	<i>ExpectedUtility</i>
<i>T</i>	<i>T</i>	$.9(.9) + .1(-.1) = .8$
<i>T</i>	<i>F</i>	$.5(.9) + .5(-.1) = .4$
<i>F</i>	<i>T</i>	$.5(.99) + .5(-.01) = .49$
<i>F</i>	<i>F</i>	$.1(.99) + .9(-.01) = .09$

Optimal: go to class and read the book.

Sequential Decision Networks

Introduction

Utility

Decision Networks

Assumption

Single-Stage

Example

Expected Utility

Sum Out

Sum Out Again

▷ Sequential
Variable Types

Example

Start Sum Out

Finish Sum Out

Start Sum Out

Finish Sum Out

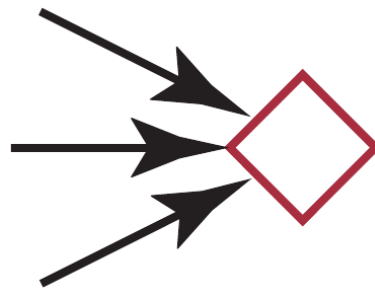
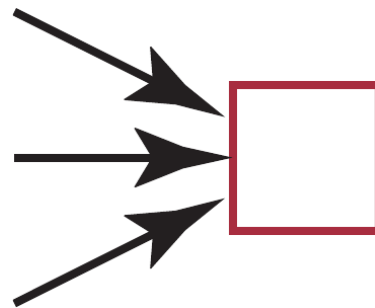
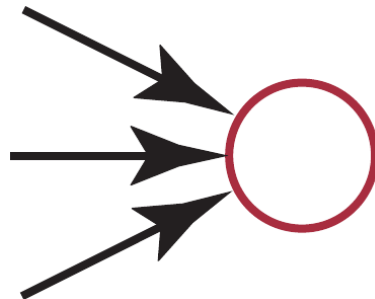
Finish Sum Out

Markov Decision
Processes

- Typically, an agent will base its decisions on information it knows.
- A *sequential decision problem* consists of a sequence of decision variables D_1, \dots, D_n .
- Each D_i has variables $parents(D_i)$, whose value will be known at the time decision D_i is made.
- A *policy* specifies what an agent should do under each circumstance. A decision function for D_i maps from values of $parents(D_i)$ to a value for D_i .
- Variable elimination: Eliminate vars below D_i , then eliminate D_i by selecting best policy.

Variable Types

- Introduction
- Utility
- Decision Networks
 - Assumption
 - Single-Stage
 - Example
 - Expected Utility
 - Sum Out
 - Sum Out Again
 - Sequential
 - ▷ Variable Types
 - Example
 - Start Sum Out
 - Finish Sum Out
 - Start Sum Out
 - Finish Sum Out
 - Finish Sum Out
 - Markov Decision Processes



- A random variable is drawn as an ellipse. Edges into the node represent probabilistic dependence.
- A decision variable is drawn as an rectangle. Edges into the node represent information available when the decision is made.
- A utility node is drawn as a diamond. Edges into the node represent variables that the utility depends on.

Example

Introduction

Utility

Decision Networks

Assumption

Single-Stage

Example

Expected Utility

Sum Out

Sum Out Again

Sequential

Variable Types

▷ Example

Start Sum Out

Finish Sum Out

Start Sum Out

Finish Sum Out

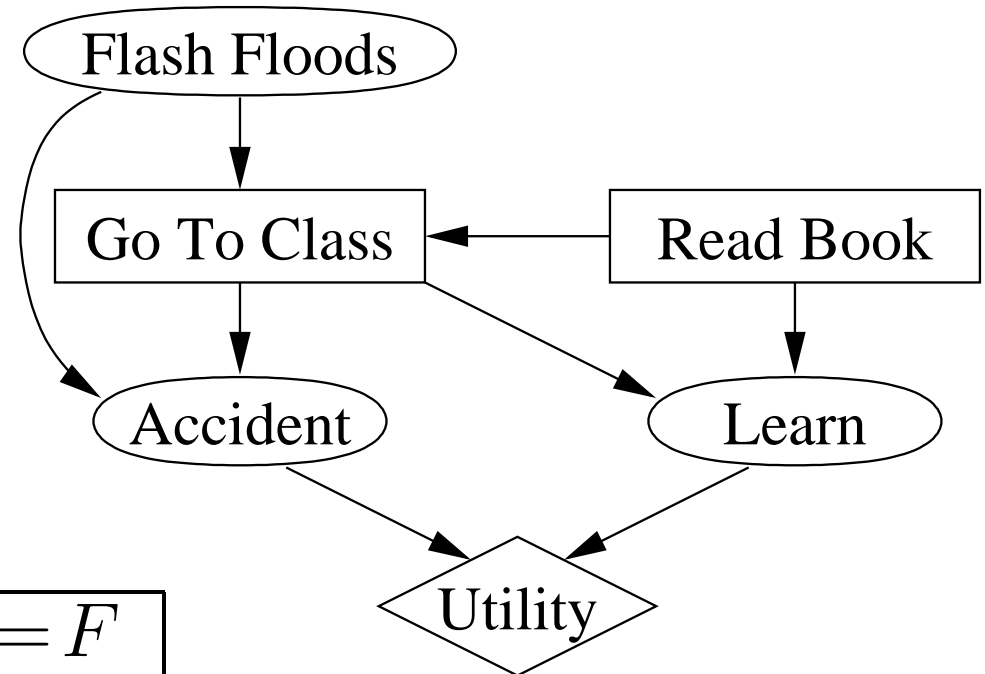
Finish Sum Out

Markov Decision

Processes

$FF = T$	$FF = F$
0.01	0.99

FF	G	$A = T$	$A = F$
T	T	0.01	0.99
T	F	0.0001	0.9999
F	T	0.001	0.999
F	F	0.0001	0.9999



Start Sum Out

Introduction

Utility

Decision Networks

Assumption

Single-Stage

Example

Expected Utility

Sum Out

Sum Out Again

Sequential

Variable Types

Example

▷ Start Sum Out

Finish Sum Out

Start Sum Out

Finish Sum Out

Finish Sum Out

Markov Decision
Processes

FF	G	A	P
T	T	T	0.01
T	T	F	0.99
T	F	T	0.0001
T	F	F	0.9999
F	T	T	0.001
F	T	F	0.999
F	F	T	0.0001
F	F	F	0.9999

*

A	L	U
T	T	-99
T	F	-100
F	T	1
F	F	0

Finish Sum Out

Introduction

Utility

Decision Networks

Assumption

Single-Stage

Example

Expected Utility

Sum Out

Sum Out Again

Sequential

Variable Types

Example

Start Sum Out

▷ Finish Sum Out

Start Sum Out

Finish Sum Out

Finish Sum Out

Markov Decision

Processes

Multiplying and summing out A yields:

FF	G	L	<i>ExpectedUtility</i>
T	T	T	$.01(-99) + .99(1) = 0$
T	T	F	$.01(-100) + .99(0) = -1$
T	F	T	$.0001(-99) + .9999(1) = .99$
T	F	F	$.0001(-100) + .9999(0) = -.01$
F	T	T	$.001(-99) + .999(1) = .9$
F	T	F	$.001(-100) + .999(0) = -.1$
F	F	T	$.0001(-99) + .9999(1) = .99$
F	F	F	$.0001(-100) + .9999(0) = -.01$

Start Other Sum Out

- Introduction
- Utility

- Decision Networks
- Assumption
- Single-Stage
- Example
- Expected Utility
- Sum Out
- Sum Out Again
- Sequential
- Variable Types
- Example
- Start Sum Out
- Finish Sum Out
- ▷ Start Sum Out
- Finish Sum Out
- Finish Sum Out

- Markov Decision Processes

<i>G</i>	<i>R</i>	<i>L</i>	<i>P</i>		<i>FF</i>	<i>G</i>	<i>L</i>	<i>EU</i>
<i>T</i>	<i>T</i>	<i>T</i>	0.9		<i>T</i>	<i>T</i>	<i>T</i>	0.0
<i>T</i>	<i>T</i>	<i>F</i>	0.1		<i>T</i>	<i>T</i>	<i>F</i>	-1.0
<i>T</i>	<i>F</i>	<i>T</i>	0.5		<i>T</i>	<i>F</i>	<i>T</i>	0.99
<i>T</i>	<i>F</i>	<i>F</i>	0.5	*	<i>T</i>	<i>F</i>	<i>F</i>	-0.01
<i>F</i>	<i>T</i>	<i>T</i>	0.5		<i>F</i>	<i>T</i>	<i>T</i>	0.9
<i>F</i>	<i>T</i>	<i>F</i>	0.5		<i>F</i>	<i>T</i>	<i>F</i>	-0.1
<i>F</i>	<i>F</i>	<i>T</i>	0.1		<i>F</i>	<i>F</i>	<i>T</i>	0.99
<i>F</i>	<i>F</i>	<i>F</i>	0.9		<i>F</i>	<i>F</i>	<i>F</i>	-0.01

Finish Sum Out

Introduction

Utility

Decision Networks

Assumption

Single-Stage

Example

Expected Utility

Sum Out

Sum Out Again

Sequential

Variable Types

Example

Start Sum Out

Finish Sum Out

Start Sum Out

▷ Finish Sum Out

Finish Sum Out

Markov Decision

Processes

Multiplying and summing out L yields:

FF	G	R	$ExpectedUtility$
T	T	T	$0.9 * (0) + 0.1 * (-1) = -0.1$
T	T	F	$0.5 * (0) + 0.5 * (-1) = -0.5$
T	F	T	$0.5 * (0.99) + 0.5 * (-0.01) = 0.49$
T	F	F	$0.1 * (0.99) + 0.9 * (-0.01) = 0.09$
F	T	T	$0.9 * (0.9) + 0.1 * (-0.1) = 0.8$
F	T	F	$0.5 * (0.9) + 0.5 * (-0.1) = 0.4$
F	F	T	$0.5 * (0.99) + 0.5 * (-0.01) = 0.49$
F	F	F	$0.1 * (0.99) + 0.9 * (-0.01) = 0.09$

Finish Sum Out

- Introduction
- Utility
- Decision Networks
- Assumption
- Single-Stage
- Example
- Expected Utility
- Sum Out
- Sum Out Again
- Sequential
- Variable Types
- Example
- Start Sum Out
- Finish Sum Out
- ▷ Finish Sum Out
- Markov Decision Processes

Choose values for G that maximize expected utility.

FF	G	R	$ExpectedUtility$
T	F	T	0.49
T	F	F	0.09
F	T	T	0.8
F	T	F	0.4

Policy for G :

Map $FF = T$ to $G = F$, and $FF = F$ to $G = T$.
Don't go to class when there are flash floods.

Policy for R : Choose $R = T$.

An MDP Game

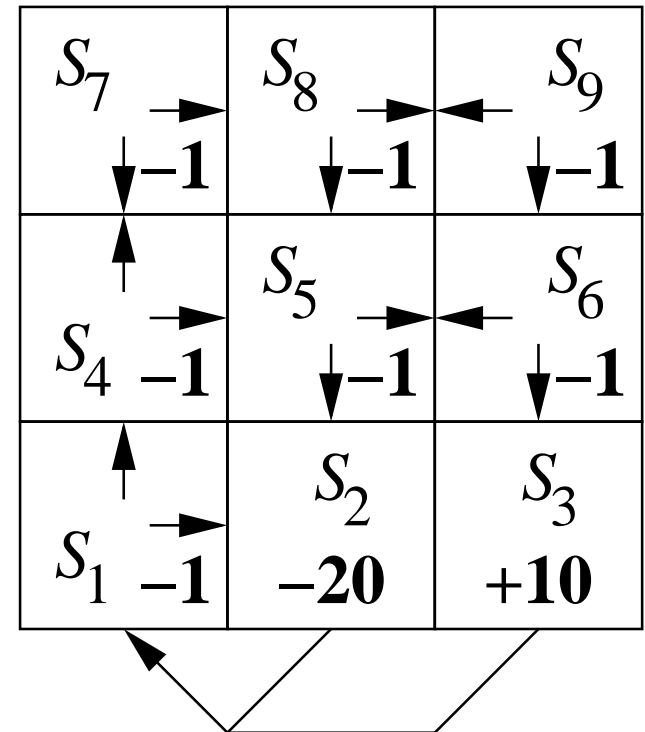
- Introduction
- Utility
- Decision Networks
- Markov Decision Processes
- ▷ An MDP Game
- Definition
- FOMDP
- POMDP
- Policies
- Policy
- Value Iteration
- Idea
- Back to MDP Game

This game has nine squares named S_1 to S_9 .

The agent starts at S_1 .

The arrows show what moves are allowed.

In S_2 and S_3 , the agent can only move to S_1 .



Each move costs 1 unit (a *reward* of -1), except S_2 (reward -20) and S_3 (reward 10).

90% of the time, the chosen move succeeds.

10% of the time, the other move happens.

Markov Decision Process

Introduction

Utility

Decision Networks

Markov Decision Processes

An MDP Game

▷ Definition

FOMDP

POMDP

Policies

Policy

Value Iteration

Idea

Back to MDP Game

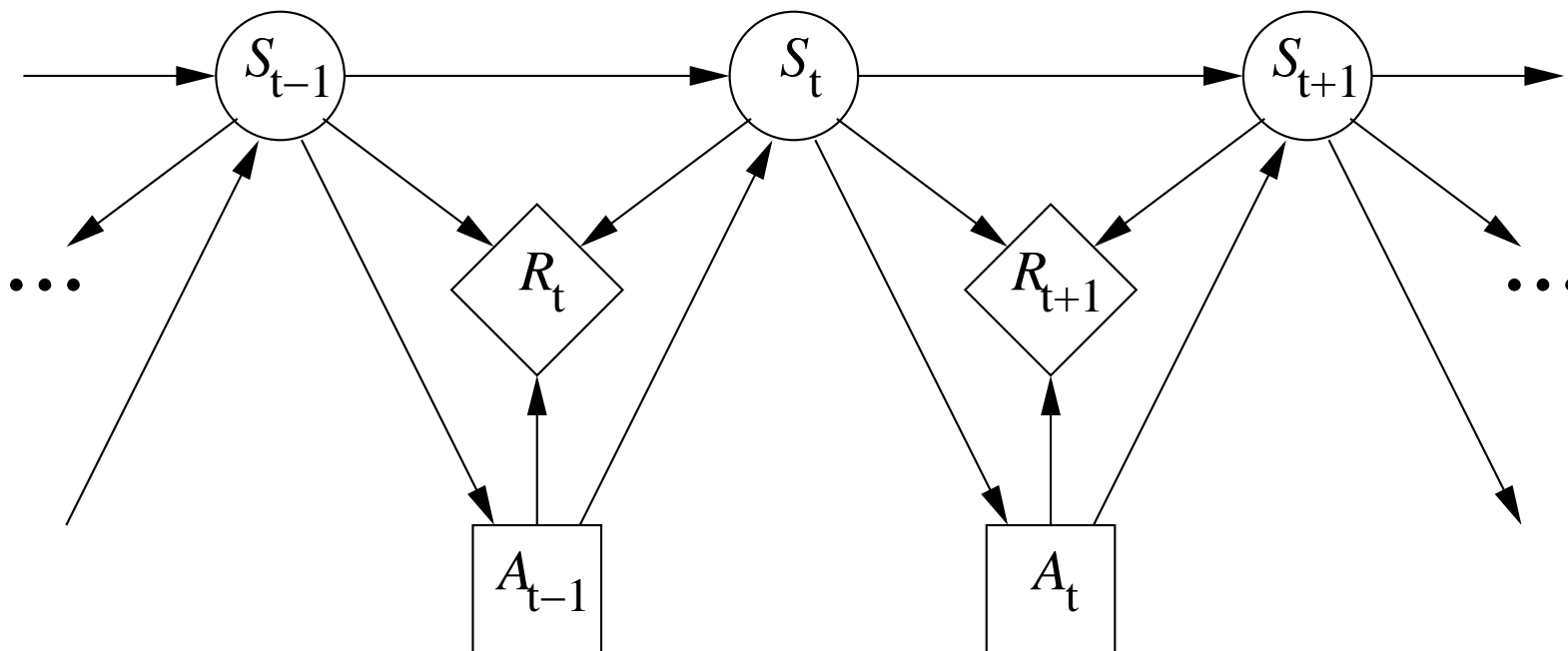
A *Markov decision process* (MDP) consists of:

- S , a set of states
- A , a set of actions
- $P : S \times S \times A \rightarrow [0, 1]$ is a probabilistic transition function $P(s' | s, a)$, the probability that action a will move from state s to state s' .
- $R : S \times A \times S \rightarrow \mathfrak{R}$ is the reward function. $R(s, a, s')$ is the reward when action a moves from state s to state s' .

The next state depends only on the current state and action. The previous states and actions do not add any more information.

Decision Network: Fully Observable MDP

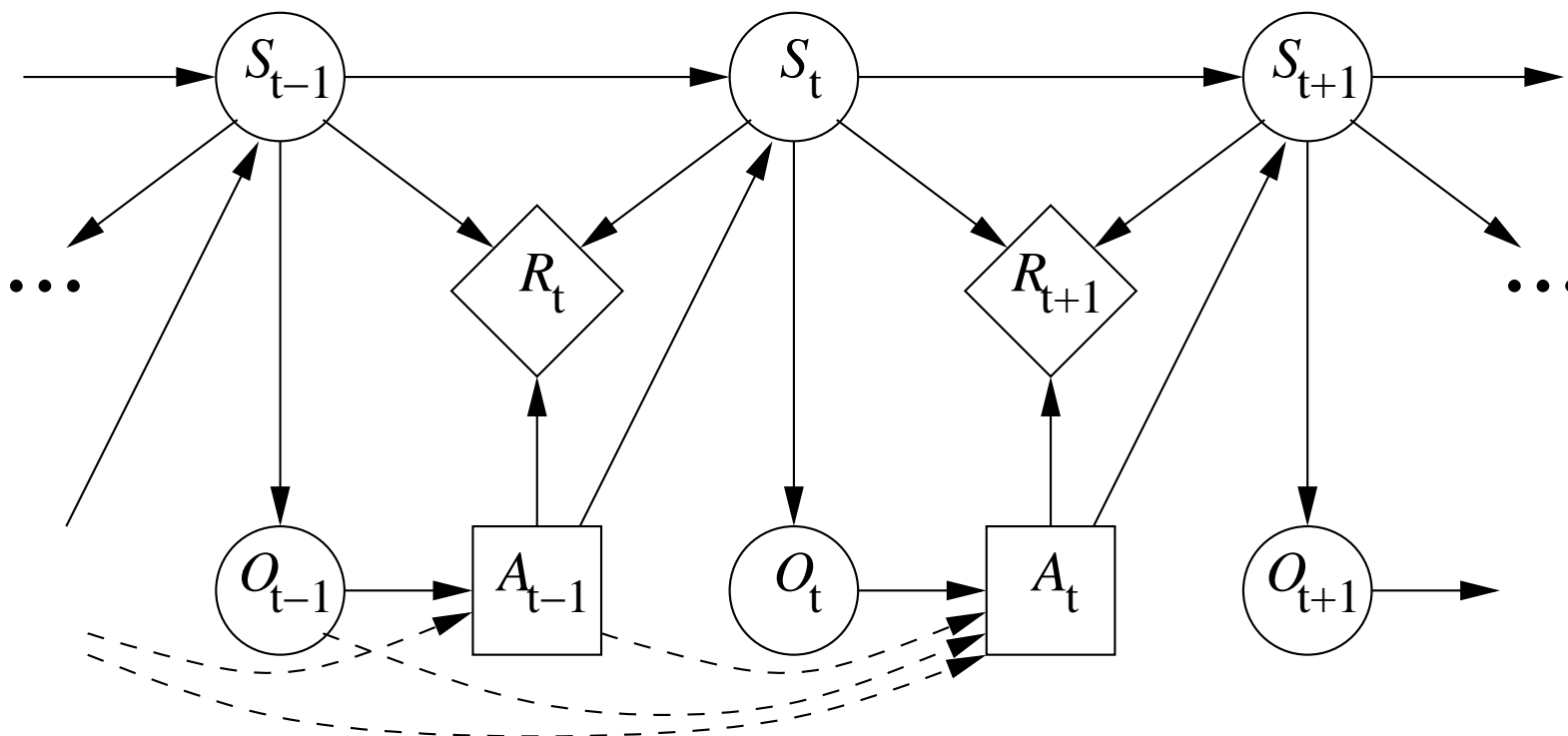
- Introduction
- Utility
- Decision Networks
- Markov Decision Processes
- An MDP Game
- Definition
- ▷ FOMDP
- POMDP
- Policies
- Policy
- Value Iteration
- Idea
- Back to MDP Game



In a *fully observable* MDP, the agent knows the current state.

Decision Network: Partially Observable MDP

- Introduction
- Utility
- Decision Networks
- Markov Decision Processes
- An MDP Game
- Definition
- FOMDP
- ▷ POMDP
- Policies
- Policy
- Value Iteration
- Idea
- Back to MDP Game



In a *partially observable* MDP (POMDP), the agent only sees evidence of the current state: current observations, previous actions/ observations, etc.

Policies

- Introduction
- Utility
- Decision Networks
- Markov Decision Processes
- An MDP Game
 - Definition
 - FOMDP
 - POMDP
 - ▷ Policies
 - Policy
- Value Iteration
- Idea
- Back to MDP Game

- A *policy* is a function: $\pi : S \rightarrow A$
- Given a state s , $\pi(s)$ specifies the action.
- An *optimal policy* is the one with the maximum value (expected discounted reward).
- For a sequence of rewards $r_1, r_2, r_3 \dots$, the *discounted reward* is:

$$V = r_1 + \gamma r_2 + \gamma^2 r_3 + \dots = \sum_{i=1}^{\infty} \gamma^{i-1} r_i$$

where $0 < \gamma < 1$ is the *discount factor*.

- For a fully-observable MDP with stationary dynamics and rewards and with infinite or indefinite horizon, there is always an optimal stationary policy.

The Value of a Policy

Introduction

Utility

Decision Networks

Markov Decision Processes

An MDP Game

Definition

FOMDP

POMDP

Policies

▷ Policy

Value Iteration

Idea

Back to MDP Game

- $Q^\pi(s, a)$ is the expected value of doing action a in state s , then following policy π .
- $V^\pi(s)$ is the expected value of following policy π in state s .
- Q^π and V^π can be recursively defined:
$$V^\pi(s) = Q^\pi(s, \pi(s))$$
$$Q^\pi(s, a) = \sum_{s'} P(s'|s, a)(R(s, a, s') + \gamma V^\pi(s'))$$
- π is an optimal stationary policy if and only if:
$$\pi(s) = \arg \max_a Q^\pi(s, a)$$
- Value iteration is an algorithm for optimizing π .

Value Iteration for FOMDPs

Introduction

Utility

Decision Networks

Markov Decision
Processes

An MDP Game

Definition

FOMDP

POMDP

Policies

Policy

▷ Value Iteration

Idea

Back to MDP Game

Procedure *Value-Iteration*(S, A, P, R, ϵ)

Inputs: states, actions, transition function,
reward function, convergence threshold

$V \leftarrow$ array with $|S|$ values

$Q \leftarrow$ 2D array with $|S| \times |A|$ values

repeat

 for each state $s \in S$

 for each action $a \in A$

$$Q[s, a] \leftarrow \sum_{s'} P(s'|s, a)(R(s, a, s') + \gamma V(s'))$$

$$V[s] \leftarrow \max_a Q[s, a]$$

until no value in V changes by more than ϵ

return V, Q

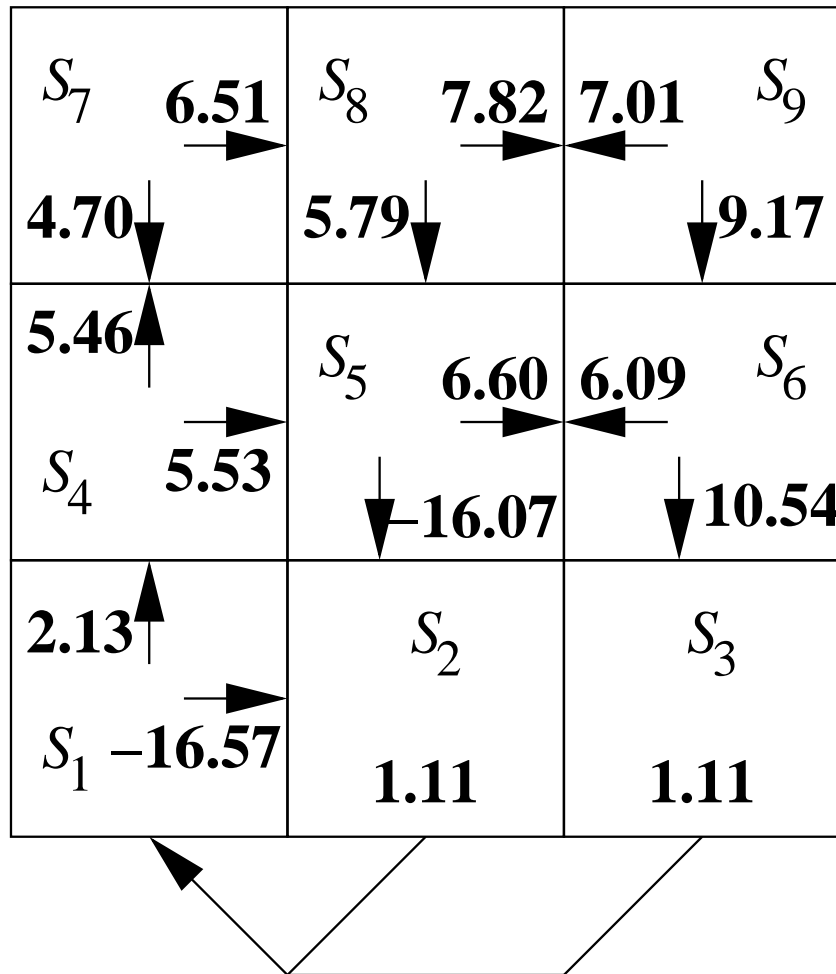
Idea of Value Iteration

- Introduction
- Utility
- Decision Networks
- Markov Decision Processes
- An MDP Game
- Definition
- FOMDP
- POMDP
- Policies
- Policy
- Value Iteration
 - ▷ Idea
- Back to MDP Game

- Assume V, Q initialized to zeroes.
- After one loop, V is the maximum expected immediate reward.
- After k loops, V is the maximum expected discounted reward, looking k steps ahead.
- It converges exponentially fast (in k) to the optimal values. The error is proportional to γ^k (assuming $\gamma > 0.5$).
- Optimal policy can be easily computed from Q or V .

Back to the MDP Game

Using $\gamma = 0.99$, $\epsilon = 0.001$, here are the Q values from value iteration.



Introduction

Utility

Decision Networks

Markov Decision Processes

An MDP Game

Definition

FOMDP

POMDP

Policies

Policy

Value Iteration

Idea

▷ Back to MDP Game